**EXCEPTION HANDLING:**

**What is Exception?**

* In our application, there are chances of abnormal condition, normal flow of program get disturbed, such situation is called as an exception.
* All types of exceptions only occurs at runtime.
* So avoiding such abnormal termination we need to handle exception.
* Exception will occur because of wrong user input.

**What is Exception Handling?**

* To avoid occurring exception we need to provide alternative way to execute rest of program, this is called as exception handling.
* Keywords to handle exception: try, catch, finally, throw, throws.

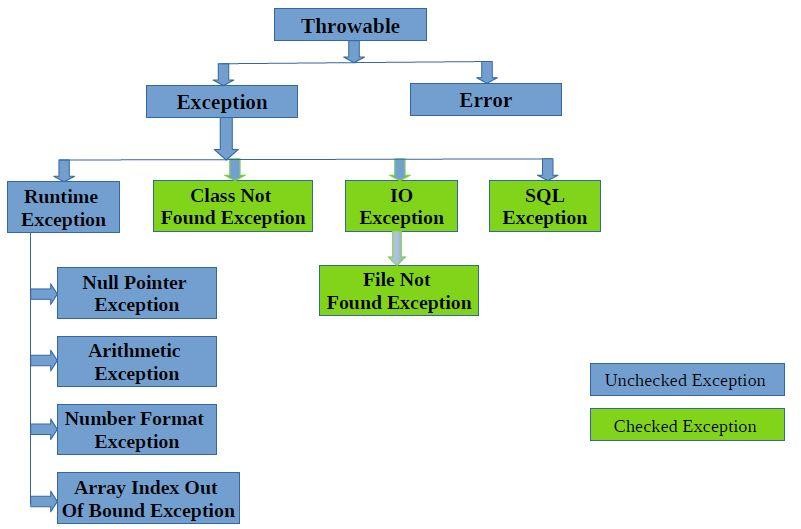
**Difference between Exception and Error?**

|  |  |
| --- | --- |
| Exception | Error |
| Classified as checked and unchecked type. | Classified as an unchecked type. |
| It belongs to java.lang.Exception. | It belongs to java.lang.error. |
| We need to handle the exceptions. | We need to solve the errors. |
| It can occur at run-time & compile-time  both. | It doesn't occur at compile-time, only  occurs at run-time. |
| E.g. NullPointerException , SqlException,  etc. | E.g. OutOfMemoryError ,  AbstractMethodError, etc. |

**Default Exception handling in Java?**

* Step-1: Problem Analysed.
* Step-2: Problem Finding.
* Step-3: Create Object of an Exception found.
* E.g. ArithmeticException e=new ArithmeticException ( );
* Step-4: Throw e ;
* Step-5: JVM will catch.
* Step-6: JVM will display Exception message.

**Explain Exception hierarchy?**



**Difference between Checked Exception & Unchecked Exception?**

|  |  |
| --- | --- |
| Checked Exception | Unchecked Exception |
| Exceptions that compiler forces user to write handling code before compilation, it  is called checked exception. | Exceptions that compiler doesn’t forces user to write handling code are called  checked exception. |
| Compulsory need to handle them. | Not compulsion to handle them. |
| It increases code. | It reduces code. |
| E.g. IOexception, SqlException,  FileNotFoundException, etc. | All Runtime Exception as well as all Errors  are example of unchecked exception. |

**What are methods in Exception class?**

* public String getMessage()
* Returns a detailed message about the exception that has occurred.
* public Throwable getCause() Returns the cause of the exception.
* public String toString()
* Returns the name of the class concatenated with the result of getMessage().
* public void printStackTrace()
* Prints the result of toString() along with the stack trace, the error output stream.
* public StackTraceElement [] getStackTrace()
* Returns an array containing each element on the stack trace.
* public Throwable fillInStackTrace()
* Fills the stack trace of this Throwable object with the current stack trace, adding to any previous information in the stack trace.

**What is use of finally block?**

“finally” block code always be executed either there is problem inside “try” block or there is no problem inside “try” block. In both situation, “finally” block code will be executed.

Resources which are open inside “try” block, those resources should be closed inside “finally” block. E.g. If database connection open then it should be closed inside “finally” block.

**Syntax :**

try {

try {

}

}

finally {

catch ( ) {

}

}

finally {

}

We cannot write two or more “finally” block for one “try” block, only one “finally” block can be written with “try” block.

We cannot write “finally” block before “catch” block, we can write it only after “catch” block like try-catch-finally.

**How to stop executing finally block?**

* Before finally block get executed, System.exit(0); line get executed.
* Before finally block get executed, unending loop get executed.

**What is final, finally, finalized?**

* The final keyword in java is used to restrict the user. The java final keyword can be used in many context. Final can be used for variables, methods, class.
* Final variable once assigned can’t be changed after.
* Final method can’t be rewritten, can’t be inherited.
* Final class can’t be accessed by creating child of it.
* The finally is a block that always be executed either there is exception occur inside “try” block or there is no exception occur inside “try” block. In both situation, “finally” block code will be executed.
* The finalize() is called by the garbage collector on an object when garbage collection determines that there are no more references to the object, it is used to perform clean- up activity.

**What is throws keyword?**

By using “throws” keyword we can give a chance to caller method to handle the exception.

The “throws” keyword is used for propagating the exception.

Whenever unchecked exception will occur, it will automatically propagated.

Whenever checked exception will occur, we need to write “throws” keyword for propagating the exception.

e.g. public void m1( ) throws IO Exception, SQL Exception

{ }

**What s throw keyword?**

The “throw” keyword is used for occurring our own exception.

To occur custom exception we use “throw” keyword.

Syntax :

public void m1( )

{

Arithmetic Exception e=new Arithmetic Exception( ) ; throw e;}

OR

public void m2( )

{

throw new Arithmetic Exception( ) ;

}

**Difference between throw & throws?**

|  |  |
| --- | --- |
| throw | throws |
| It is used to occur custom exception. | It is used to give a chance to caller method  to handle the exception. |
| This keyword is used while creating new instance of exception class. | This keyword is used with method name to declare the exception that need to  propagate. |
| Can be used only with single exception at a  time. | Can be used with multiple exceptions at a  time. |
| You cannot throw multiple exceptions. | You can declare multiple exceptions |

**Why to create Customise Exception?**

* Customise Exception tells user user-friendly message, as user can’t understand programming language code for exception.

**Customise Exception for any Checked Exception?**

* First create user-define class for customise exception, extend it to corresponding checked exception class, and also create parameterised constructor that accepts string input and forward that variable using super keyword.
* Now use throw keyword and create new instance of custom class you have created where you want to handle checked exception in your application.

**Can we create Customise Exception for Checked Exception?**

* Yes we can but not necessary, because customise exception used for telling user user- friendly message as user doesn’t know programming language.
* But programmer is aware of programming codes, so programmer doesn’t require customise exception.

**New Exception related feature in JDK 1.7 version?**

* Try with Resource.

**Which type of statements can be written in try with resources?**

* We can write those classes/interfaces which have implemented/extended Autoclosable interface only.
* For user-defined class/interface, we need to implement/extend Autoclosable interface explicitly. So that we can write that class in try with resources context.

**Difference between ClassCasteException and NoClassDefFound Error?**

|  |  |
| --- | --- |
| ClassCasteException | NoClassDefFound Error |
| It occurs when we are trying to type cast  parent object to Child. | It occurs when it is unable to find required  .class file at runtime. |
| It can be avoided by correcting type while  type casting, by changing in code. | It can be avoided by providing required  .class file. |
| It is child class of Runtime Exception. | It is child class Error. |

**Different Scenarios in Exception Handling?**

* + Once problem will occur inside “try” block, then remaining line of code of “try” block will not be executed.
* If there is no problem inside “try” block, then “catch” block will be skipped. In “catch” block we can write same class of exception and its parent classes only.
* We can write two or more “catch” for one “try” block, but we can write child classes of exception in first catch blocks and then parent class exception, we cannot write first parent exception before child exception.
  + If “try” with “finally” block and “try” have return statement, then before control return back to caller first of all “finally” block code will be executed.
* If “try” with “finally” block, “try” have return statement and if “finally” block change return statement value, then it will be changed only for “finally” block, it will not be changed for caller.
* If “try” and “finally” both have return statement, then “finally” block statement will be executed.
* “throws” with method calling, in case of unchecked exception classes :
  + For unchecked exceptions, it is not needed to write explicitly throws exception, the exception propagation is done implicitly to the caller.
* In case of checked exception classes, needed to write explicitly throws exception for propagating exception to the caller.
* “throws” keyword with method Overriding:
* (In case of unchecked exceptions) If Parent class method throws unchecked
* exception or doesn’t throws any exception then at the time of overriding no need to
* write throws keyword, if we want we can write ay of unchecked exception class but not checked exception class.
* (In case of checked exceptions) If Parent class method throws checked exception then at the time of overriding no need to write throws keyword, if we want then we can write same exception class or their child class but not parent class.
* “throws” keyword with Super Class constructor and Sub Class constructor:
* (For unchecked exception) If Parent class constructor doesn’t throws any exception or “throws” unchecked exception then child class constructor no need to write
* “throws keyword, if we want then any of exception class they can write.
* (For checked exception) If parent class constructor throws exception then at a time of inheritance child class constructor must write “throws” keyword with same exception class or their parent class.